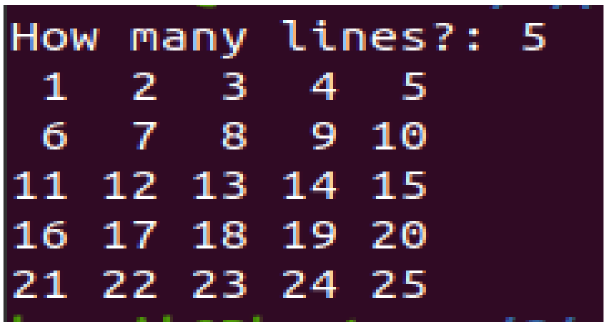
1. Write a Program to get an output as shown below.

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

```
#include<stdio.h>                        //inclusion of header file
int main()                               //main function declaration
{
      int n, i, j, count = 1;            //variable declaration
      printf("How many lines?: ");       //Asking input from user
      scanf("%d", &n);

      for (i = 1; i <= n; i++)           //outer loop
{
            for (j = 1; j <= n; j++)     //inner loop
{
                  printf("%2d ", count++); //inner loop statement
            }
            printf("\n");                //outer loop statement
      }
      return 0;
}
```



```
How many lines?: 5
 1  2  3  4  5
 6  7  8  9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

Nesting of loops or nested loops is a concept that allows the looping of statements inside another loop. In the above program, **for** loop is nested inside an outer **for** loop. Any number of loops can be defined inside another loop.

Syntax of nested loop:

```
Outer_Loop{
            Inner_Loop{
                  //inner loop statements
            }
            //outer loop statements
}
```

Outer_Loop & **Inner_Loop** are valid loops that can be a '**for', 'while',** or **'do while'** loop.

The above program uses nested loop to ask a user about how many rows and columns they want to display and the program generates a pattern as shown.

Illustration:
If a user inputs 5 then the outer loop runs until n = 5 , so does the inner loop.
At first, value of i = 1 , which is less than n = 5. Thus, the loop enters inner loop until the condition is false (i.e. until j = 5). Until then the value of count is printed increasing it by one until the condition is satisfied.

i.e.   1 2 3 4 5

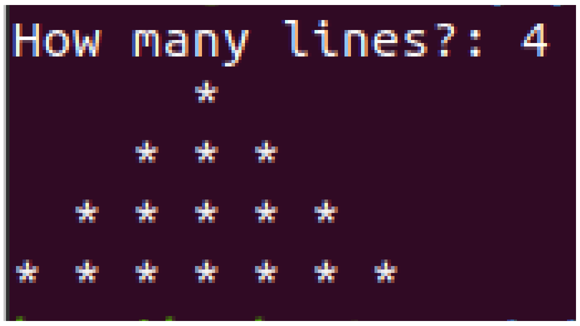
is printed.
After that the inner loop breaks and I is increased by 1 , thus i = 2, and again the loop is executed as above until value of i becomes 5. After that the pattern as shown in the picture is printed and the loop breaks and the program exits.

2. Write a program to get an output as shown in figure.

```
      *
     ***
    *****
   *******
```

```
#include<stdio.h>
int main()
{
	int n, i, j=0, k;		//variable declaration

	printf("How many lines?: ");	//asking user input
	scanf("%d", &n);
	for (i = 1; i <= n; i++)	//outer loop
	{
	//inner loop for printing space (blank character)
		for (k = 1; k <= (n-i); k++)
		{
			printf("  ");
		}
	//inner loop for printing (*) pattern
		for (j = 1; j <= (2 * i -1); j++)
		{
			printf("* ");
		}
	printf("\n"); //outer loop for changing line
	}
	return 0;
}
```



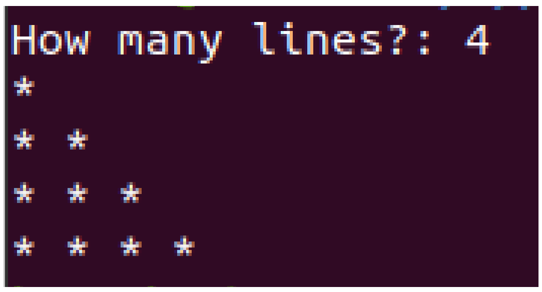
```
How many lines?: 4
      *
    * * *
  * * * * *
* * * * * * *
```

This is a program that asks user the number of lines of pyramid pattern (*) they want to display and then displays it in the screen. This program also uses the concept of nested looping but in this case two separate for loops are nested inside a common outer for loop.
The first loop displays space(blank character) until k becomes equal to (**n – i)** i.e. if n = 4, then k loops 3 times.
i.e. In first line, **three** blank space is printed, in second line, **two** blank space is printed, after that **one** blank space is printed and finally no blank space is printed.
The other for loop containing **j variable** is used for displaying pyramid pattern. The loop runs until j becomes equal to (**2 x i – 1)** where i loops until it becomes equal to **n = 4.**
The final result is as displayed in the picture above.

3. Write a Program to get an output as shown in figure.

```
*
* *
* * *
* * * *
```

```
#include<stdio.h>
int main()
{
      int n, i, j;                      //variable declaration
      printf("How many lines?: ");      //asking user input
      scanf("%d", &n);
      for (i = 1; i <= n; i++)          //outer for loop
      {
            for (j = 1; j <= i; j++)    //inner for loop
            {
                  printf("* ");         //inner loop statement
            }
            printf("\n");               //outer loop statement
      }
return 0;
}
```



```
How many lines?: 4
*
*  *
*  *  *
*  *  *  *
```

This is a program that asks user how many lines of pattern they want to generate and displays the output as shown in the picture. This program also uses the concept of nested looping.
Illustration:
Suppose the user inputs the value of 5. i.e. n = 5
Then the outer loop containing the variable **i loops** until it becomes equal to 5,
Thus until the condition is true, the loop enters inner for loop containing the variable **j.**
At first, j loops until it becomes equal to i, in this case 1.

Thus first line (*) is printed. And the loop exits and line is changed. Value of I is increased by one. i.e. I = 2 now
Again the program enters inner loop and it is looped until j = 2
Thus next line (* *) is printed and the process is repeated until I = 5 , generating the required pattern and then the loop breaks, the program exits.

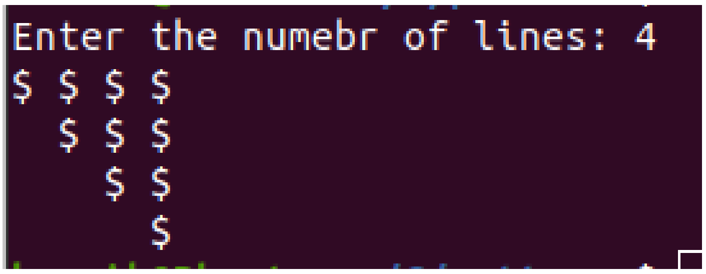
4. Write program to get an output as shown in figure.

```
$ $ $ $
$ $ $
$ $
$
```

```
#include<stdio.h>
int main()
{
     int n, i, k, j;                          //variable declaration
     printf("Enter the numebr of lines: ");//asking user input
     scanf("%d", &n);
     for (i = n; i > 0; i--)                  //outer for loop
     {

     //inner for loop for space printing
          for (k = 1; k <= (n-i); k++)
          {
               printf("  ");                  //inner loop_1 statement
          }
     //inner for loop for printing $ pattern
          for (j = 1; j <= i; j++)
          {
               printf("$ ");                  //inner loop_2 statement
          }
          printf("\n");                       //outer loop statement

     }
     return 0;
}
```



```
Enter the numebr of lines: 4
$ $ $ $
  $ $ $
    $ $
      $
```

This is a program that asks user the number of lines of pattern they want to display and displays the pattern accordingly as shown in the picture. This program also uses the concept of nested looping and as in question number 2, it uses two separate for loops inside a common outer for loop.
The iteration of outer loop begins from ‘n’ and subsequently decreases until it becomes equal to 1.

The inner loop containing the variable **k** loops until (n – i) times.
Illustration:
Consider user input n = 4.
Thus I = 4 , at first.
K loops until n – I times which is equal to 0 in this case. Thus this loop doesn’t execute at first.
The loop containing **j** variable loops from 1 to I , i.e. 1 to 4 in first case.
And thus ($ $ $ $) is printed and the loop breaks and line changes
Value of I is decreased by 1, thus new I = 3, and the looping continues
k loops until n – I, which is equal to 4 – 3 = 1 times. Thus , it prints one blank space this time.
J loops until 1 to 3 this time
Thus it prints ($ $ $) this time.
The line is changed again
The loop is executed until I = 1, and then the loop breaks and the program exits generating the above pattern.

KATHMANDU UNIVERSITY

DHULIKHEL, KAVRE

SUBJECT : ……………………..

LAB SHEET NO : …………….

<u>SUBMITTED BY:</u>	**<u>SUBMITTED TO:</u>**
NAME :	
ROLL NO. :	Department of
GROUP :	
LEVEL : …. /....	